Reconfiguration dans les réseaux optiques †

D. Coudert[1] and F. Huc[2] and D. Mazauric[1] and N. Nisse[1] and J.-S. Sereni[3]

1- MASCOTTE, INRIA, I3S, CNRS, Univ. Nice Sophia, Sophia Antipolis, France
2- TCS-sensor lab, Centre Universitaire d'Informatique, Université de Genève, Suisse
3- CNRS (LIAFA, Univ. D. Diderot), Paris et KAM (Fac. of Math. and Physics), Charles Univ., Prague, Rép. Tchèque

L'évolution permanente du trafic, les opérations de maintenance et l'existence de pannes dans les réseaux WDM, obligent à rerouter régulièrement des connexions. Les nouvelles demandes de connexions sont routées en utilisant les ressources disponibles et, si possible, sans modifier le routage des connexions existantes. Ceci peut engendrer une mauvaise utilisation des ressources disponibles. Il est donc préférable de reconfigurer régulièrement l'ensemble des routes des différentes connexions. Un objectif particulièrement important est alors de minimiser le nombre de requêtes simultanément interrompues lors de la reconfiguration. Nous proposons une heuristique pour résoudre ce problème dans les réseaux WDM. Les simulations montrent que cette heuristique réalise de meilleures performances que celle proposée par Jose et Somani (2003). Nous proposons également un modèle permettant de prendre en compte différentes classes de clients, avec notamment la contrainte que des requêtes, dites prioritaires, ne peuvent pas être interrompues. Une simple transformation permet de réduire le problème avec requêtes prioritaires au problème initial. De ce fait, notre heuristique s'applique également au cas autorisant des requêtes prioritaires.

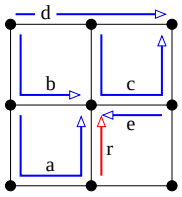
1 Introduction et modèle

Optimiser l'utilisation des ressources d'un réseau constitue un enjeu critique pour les compagnies de télécommunications exploitant des réseaux WDM. La demande de trafic augmente de manière importante (par exemple, la Chine voit apparaître 6 millions de nouveaux utilisateurs d'Internet chaque mois) et ce trafic est sujet à une variation constante en raison du déploiement de nouveaux services. C'est pourquoi le routage de ce trafic doit être mis à jour régulièrement pour assurer une utilisation efficace des ressources des réseaux et pour préserver une flexibilité suffisante pour accueillir de nouvelles requêtes rapidement. Par exemple, la Figure 1(a) représente un réseau en grille utilisant une longueur d'onde (une dans chaque sens) au sein duquel cinq connexions a, b, c, d et e sont initialement établies. La nouvelle requête r ne peut pas être acceptée (r et a sont en conflit) alors que le routage représenté par la Figure 1(b) est possible. Pour réduire la probabilité d'inter-blocage [LXC05], les routes empruntées par certaines requêtes doivent être modifiées, et donc le routage doit être reconfiguré [JS03]. La tolérance aux pannes est un autre enjeu important des réseaux. En effet, des pannes dans le réseau backbone peuvent avoir un impact important avec des répercussions financières graves. C'est pourquoi des mécanismes de restauration et de protection du trafic sont mis en place pour assurer la continuité du trafic lorsque des pannes se produisent. Ces mécanismes sont rapides et assurent de faibles perturbations du trafic. Cependant, la durée des actions entreprises sur le réseau (réparations par exemple) peuvent être longues (plusieurs jours). Ainsi ces actions peuvent être planifiées de façon à optimiser l'utilisation des ressources, c'est-à-dire, reconfigurer le routage.

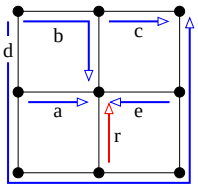
Dans cet article, nous nous concentrons sur le problème de reconfiguration du routage: modifier les routes d'un ensemble de connexions pour passer du routage courant à un routage destination pré-calculé, sous la contrainte de changer les routes des requêtes les unes après les autres (le calcul du routage destination n'est pas pris en compte ici).

1.1. Graphe de dépendance. Pour modifier la route empruntée par une requête établie, il faut s'assurer que les ressources de la route de destination soient libres. Dans l'exemple de la Figure 1, la connexion b doit être déplacée avant l'établissement de la requête a car cette dernière utilise, dans le nouveau routage R', une ressource utilisée par b dans le routage initial R. Pour modéliser ces dépendances, nous utilisons la notion

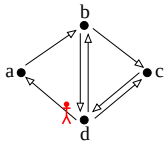
†Les auteurs ont reçu le support du projet IST AEOLUS et de la région PACA. [CHM+09] est une version étendue de cet article.



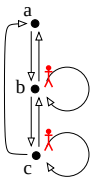
(a) Routage R où r ne peut pas être acceptée.



(b) Routage R' des requêtes a, b, c, d, e, r.



(c) Graphe de dépendance D: $pn(D) = 1$.



(d) D^* si d est prioritaire: $pn(D^*) = 2$.

Fig. 1: Exemple de reconfiguration dans un réseau en grille et graphes de dépendance correspondants.

de graphe orienté de dépendance [JS03]. Étant donné un routage initial R et un routage de destination R', le *graphe de dépendance* est composé d'un sommet par requête reroutée, et il y a un arc d'un sommet u à un sommet v si la route empruntée par la requête u dans R' utilise des ressources utilisées par v dans R. Dans l'exemple de la Figure 1, le graphe de dépendance D est composé de quatre sommets a, b, c et d (Fig. 1(c)). Il y a un arc de a vers b car a utilise des ressources dans le nouveau routage R', utilisées par b dans le routage initial R. En d'autres termes, b doit être reroutée avant a. Lorsque le graphe de dépendance est un DAG (Directed Acyclic Graph), l'ordonnancement des déplacements des connexions est immédiat. Cependant des dépendances cycliques peuvent exister. Dans ce cas, la reconfiguration n'est possible que si l'on s'autorise à interrompre temporairement des connexions. L'objectif est alors de minimiser le nombre de connexions interrompues simultanément. Ce problème a tout d'abord été étudié dans [JS03] puis a été modélisé en termes de *process number* [CPPS05] similaire à des jeux de capture (voir [FT08]).

1.2. Process Number. Dans ce cadre, le problème est défini comme un jeu au cours duquel des agents sont successivement placés et supprimés des sommets du graphe de dépendance D. Dans ce jeu, un agent placé sur un sommet de D représente l'interruption de la connexion correspondante. Un sommet $v \in V(D)$ peut être *traité* (signifiant que la requête correspondante est déplacée) lorsque chacun de ses voisins sortants $w \in N^+(v)$ est soit occupé par un agent, soit déjà traité. En d'autres termes, une connexion peut être déplacée lorsque chacune des connexions qui utilisaient les ressources nécessaires a été soit déplacée, soit interrompue. Lorsqu'un sommet occupé par un agent est traité, l'agent correspondant est supprimé du sommet et peut être réutilisé par la suite. Le *process number* $pn(D)$ d'un graphe orienté D est le nombre minimum d'agents nécessaires pour pouvoir traiter tous ses sommets. Le *process number* d'un graphe de dépendance est donc égal au nombre minimum de requêtes qui doivent être simultanément interrompues durant la reconfiguration correspondante. Par exemple, $pn(D) = 0$ si et seulement si D est un DAG (les feuilles peuvent être simultanément traitées). Le *process number* du graphe de la Figure 1(c) est 1 car en plaçant un agent sur d, nous pouvons traiter séquentiellement les sommets c, b, a et finir par le sommet d (la requête d est temporairement interrompue). Utilisant cette formulation, [CPPS05] prouve que le problème de déterminer le nombre minimum de requêtes devant être interrompues simultanément au cours du reroutage est NP-complet. Les graphes orientés de *process number* 1 ou 2 sont caractérisés dans [CS07].

1.3. Résultats. Nous proposons une heuristique $\mathcal{H}$, basée sur une méthode de flots, pour déterminer des stratégies de reconfiguration pour passer d'un routage quelconque à un autre. Nous avons validé $\mathcal{H}$ sur des graphes de dépendance avec diverses topologies. L'heuristique $\mathcal{H}$ donne des résultats significativement meilleurs que ceux de l'heuristique de Jose et Somani [JS03] même s'il est indispensable de souligner que leur but était, entre autres, de minimiser le nombre total de connexions interrompues durant la reconfiguration (heuristique pour le *Minimum Feedback Vertex Set*). $\mathcal{H}$ donne également des résultats proches de l'optimum dans des classes de graphes de *process number* connu, comme les graphes de *process number* 1 et 2 [CS07] ou les graphes d'intervalles circulaires [ST07]. Ces derniers sont particulièrement intéressants car ils correspondent aux graphes de dépendance du reroutage dans les réseaux physiques en anneau.

Nous proposons aussi un modèle pour le problème de reconfiguration lorsque certaines requêtes, dites *requêtes prioritaires*, ne peuvent pas être interrompues. Certaines configurations des requêtes prioritaires, détectables en temps linéaire, interdisent toute reconfiguration. Lorsque de telles configurations n'apparaissent pas, nous prouvons que le problème de reconfiguration avec requêtes prioritaires peut être réduit par une simple transformation au problème initial, i.e., sans requêtes prioritaires.

2 Heuristique

Dans cette partie, nous présentons une heuristique $\mathcal{H}$ calculant un ordonnancement des déplacements des requêtes pour passer d'un routage R à R' dans le but de minimiser le nombre maximum de requêtes interrompues simultanément. Plus exactement, considérant la formalisation en terme de *process number*, $\mathcal{H}$ calcule une stratégie pour traiter les n sommets d'un graphe orienté $D = (V,A)$ à m arcs.

2.1. Principes. Le *process number* d'un graphe (orienté) de dépendance est le *process number* maximum parmi ses composantes fortement connexes [CPPS05]. Ainsi $\mathcal{H}$ s'applique séquentiellement à chacune des composantes. A chaque étape, un sommet $v \in V(D)$ qui n'est ni traité ni occupé par un agent est choisi. Un agent est alors placé sur v et tous les sommets pouvant être traités le sont. L'agent occupant v est supprimé sitôt que v est traité. L'heuristique repose donc sur le choix du prochain sommet sur lequel un agent va être placé. Ce choix est le résultat d'une élection parmi les sommets ni traités ni occupés par un agent. Chaque sommet (requête) veut être traité (déplacée) le plus rapidement possible sans être occupé par un agent (interrompue). Egoïstement, un sommet a donc intérêt à ce qu'un agent soit placé sur un de ses voisins sortants. Nous proposons une méthode de circulation de flot pour modéliser cette élection à la fin de laquelle le sommet choisi est celui qui maximise le contentement des sommets du graphe.

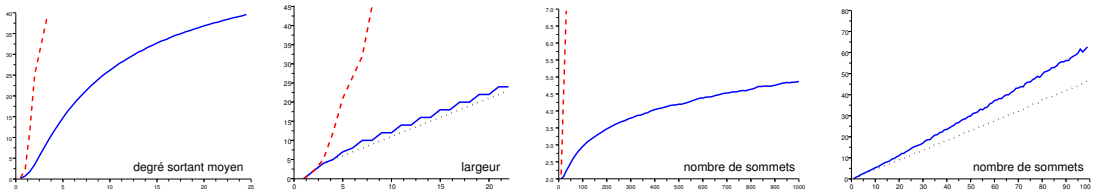
Plus formellement, à chaque étape, divisée en t rondes, $\mathcal{H}$ considère une composante connexe maximale C de D induite par des sommets ni occupés ni traités. Un poids uniforme est assigné à chacun des sommets de C et un flot circule ainsi : à chaque ronde, chaque sommet u de C envoie une fraction uniforme de son poids à chacun de ses voisins sortants, et met à jour son nouveau poids comme la somme des flots qu'il a reçu de ses voisins. Après t rondes, un agent est placé sur le sommet ayant le poids le plus élevé. En modélisant ce processus par une chaîne de Markov discrète, [CM08] prouve que le vecteur des poids admet toujours une unique solution stationnaire et que le processus converge en au plus $t = n$ rondes. L'heuristique s'exécute en temps $O(n^2(n+m))$. Ceci rend possible son utilisation pour des graphes de grande taille (et donc de grand réseau) contrairement à l'heuristique présentée dans [JS03] de complexité exponentielle.

En fait, l'heuristique définie ci-dessus peut impliquer le placement d'agents sur des sommets qui ne le nécessitent pas. Pour palier ce problème, après chaque placement d'un agent sur un sommet, si (et seulement si) des sommets peuvent être traités, nous supprimons certains agents inutiles (pour plus de précisions, voir [CHM$^+$09]). Ceci n'augmente la complexité de notre algorithme que d'un facteur n.

2.2. Performances. Pour valider l'heuristique proposée ci-dessus, nous l'avons simulé sur différentes topologies. Pour chacune de ces topologies et pour un panel de tailles ou de degrés moyens, nous créons 1000 instances aléatoires de graphes appartenant à cette topologie. Ensuite $\mathcal{H}$ est exécutée sur ces instances. Les graphiques représentent la moyenne des résultats (nombre d'agents utilisés) obtenus sur ces instances. La première topologie que nous étudions est celle des graphes aléatoires. Dans ce cas, Les résultats que nous obtenons sont présentés sur la Figure 2(a) pour un graphe orienté de 50 sommets avec 1000 instances aléatoires pour chaque degré moyen. Les performances de l'heuristique de [JS03] sont représentées à titre de comparaison. Comme le problème de déterminer le *process number* est NP-complet en général [CPPS05], il est intéressant d'étudier les performances de notre heuristique dans des classes de graphes pour lesquelles il existe des algorithmes exacts ou de bonnes approximations pour calculer ce paramètre. Dans le cas de grilles carrées (Fig. 2(b)), nos résultats sont significativement meilleurs que ceux de [JS03]. Les résultats obtenus dans la classe des graphes de *process number* 2 [CS07] sont représentés sur la Figure 2(c) en effectuant 1000 instances aléatoires pour chaque nombre de sommets. Dans ce cas, $\mathcal{H}$ calcule des stratégies utilisant 5 agents en moyenne pour des graphes de 1000 sommets. Les graphes d'intervalles circulaires sont particulièrement intéressants puisqu'ils correspondent aux graphes de dépendance obtenus lorsque le réseau physique est un anneau. Pour comparer nos résultats (Fig. 2(d)), nous avons implémenté l'algorithme de [ST07] calculant la *pathwidth* de ces graphes (la *pathwidth* d'un graphe non orienté fournit une approximation à 1 près du *process number* [CPPS05] du graphe orienté symétrique correspondant).

3 Prise en compte de requêtes prioritaires

3.1. Différentes classes de clients. La reconfiguration du routage permet à l'opérateur du réseau d'optimiser l'utilisation des ressources. Cependant, ce processus peut forcer à interrompre temporairement certaines connexions, ce qui peut entraîner des perturbations de trafic inacceptables pour certains clients. En effet



(a) Graphes aléatoires. (b) Grille carrée. (c) 2-digraphes. (d) Intervalles circulaires.

Fig. 2: Résultats de simulations : $\mathcal{H}$ (trait plein), Jose et Somani (tirets), valeur exacte (pointillés).

ces derniers peuvent avoir contracté des clauses spécifiques interdisant toute interruption de trafic. C'est pourquoi nous introduisons une nouvelle contrainte qui impose que certaines connexions, les *connexions prioritaires*, ne soient pas interrompues. Dans la formulation en termes de *process number*, cette contrainte est modélisée par une classe particulière $\mathcal{P}$ de sommets du graphe de dépendance, appelés *sommets prioritaires*. Ces sommets ne peuvent pas être occupés par des agents. Pour traiter un tel sommet, la seule solution est donc de traiter tout son voisinage sortant au préalable. Si les sommets de $\mathcal{P}$ induisent un circuit dans le graphe de dépendance, la reconfiguration est impossible [CHM$^+$09]. Etant donné un graphe D le nombre minimum d'agents nécessaires pour traiter les sommets de D avec $P \subseteq V(D)$ l'ensemble de sommets prioritaires de D est noté $pn(D,P)$. Le paramètre $pn(D,P)$ n'est donc défini que si P induit un DAG dans D. Par exemple si dans l'exemple de la Figure 1, les sommets c et d sont prioritaires, alors le graphe de dépendance D n'est pas admissible car des deux sommets forment un circuit. Notons que le *process number* d'un graphe orienté peut augmenter lorsque l'on introduit des sommets prioritaires (Figure 1(d)).

3.2. Résultats. Nous prouvons que lorsque la reconfiguration est possible, le problème de reconfiguration d'un routage avec requêtes prioritaires est équivalent au même problème en l'absence de requêtes prioritaires. Considérons la transformation suivante. Soit D un graphe orienté avec $P \subseteq V(D)$ l'ensemble des sommets prioritaires. Pour tout $v \in P$, supprimer v et ajouter un arc de chaque $w \in N^-(v)$ à chaque $w' \in N^+(v)$. Soit D^* le graphe résultant. Un tel graphe D^* obtenu par transformation de D avec $P = \{d\}$ est représenté par la Figure 1(d). Par manque de place, la preuve du théorème suivant est omise (voir [CHM$^+$09]).
Théorème. Soit D un graphe orienté et $P \subseteq V(D)$. Si P induit un DAG alors $pn(D^*) = pn(D,P)$.

4 Conclusion

La formalisation en termes de *process number* offre d'intéressantes perspectives pour la reconfiguration efficace des routages dans les réseaux WDM. Plusieurs généralisations sont à envisager. Par exemple, lorsque les liens du réseau physique peuvent être partagés par différentes routes ou lorsque plus de deux classes de services (clients) sont à prendre en compte. Par ailleurs, il serait intéressant de valider notre heuristique dans le cas de graphes de dépendance issus d'instances réelles de routage.

References

[CHM$^+$09] D. Coudert, F. Huc, D. Mazauric, N. Nisse, and J.-S. Sereni. Routing reconfiguration/process number: Coping with two classes of services. In 13th *Conf. on Optical Network Design and Modeling*, 2009.

[CM08] D. Coudert and D. Mazauric. Network reconfiguration using cops-and-robber games. Technical Report inria-00315568, INRIA, August 2008.

[CPPS05] D. Coudert, S. Perennes, Q.-C. Pham, and J.-S. Sereni. Rerouting requests in wdm networks. In *AlgoTel'05*, pages 17–20, mai 2005.

[CS07] D. Coudert and J-S. Sereni. Characterization of graphs and digraphs with small process number. Research Report 6285, INRIA, September 2007.

[FT08] F. Fomin and D. Thilikos. An annotated bibliography on guaranteed graph searching. *TCS*, 399(3):236–245, 2008.

[JS03] N. Jose and A.K. Somani. Connection rerouting/network reconfiguration. *Design of Reliable Communication Networks*, pages 23–30, October 2003.

[LXC05] K. Lu, G. Xiao, and I. Chlamtac. Analysis of blocking probability for distributed lightpath establishment in WDM optical networks. *IEEE/ACM Trans. Netw.*, 13(1):187–197, February 2005.

[ST07] K. Suchan and I. Todinca. Pathwidth of circular-arc graphs. In *Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 258–269, 2007.